A PRELIMENERY REPORT ON

BRAIN TUMOR SEGMENTATION

SUBMITTED TO THE VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY (COMPUTER ENGINEERING)

SUBMITTED BY

ABHISHEK ATHANIKAR 22111031
YASHRAJ AWARE 22110167
YASH BORUDE 22110013
RAVIRAJ CHOUGULE 22111093

DEPARTMENT OF COMPUTER ENGINEERING

BRACT'S
VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY SURVEY NO.

3/4, KONDHWA (BUDRUK), PUNE – 411048, MAHARASHTRA (INDIA).

\

1. INTRODUCTION

Alzheimer’s disease (AD) is a chronic neurodegenerative disease mainly occurring in the elderly. As Brain tumor segmentation plays a crucial role in medical image analysis, aiding clinicians in diagnosis, treatment planning, and monitoring of brain disorders. Accurate segmentation of tumors from magnetic resonance imaging (MRI) scans is essential for precise localization and characterization of tumors. Automated segmentation methods can streamline this process, saving time and improving accuracy.

This report outlines a project focused on brain tumor segmentation using deep learning techniques. The goal is to develop an efficient and accurate segmentation model capable of delineating tumor regions from brain MRI scans.

2. LITERATURE SURVEY

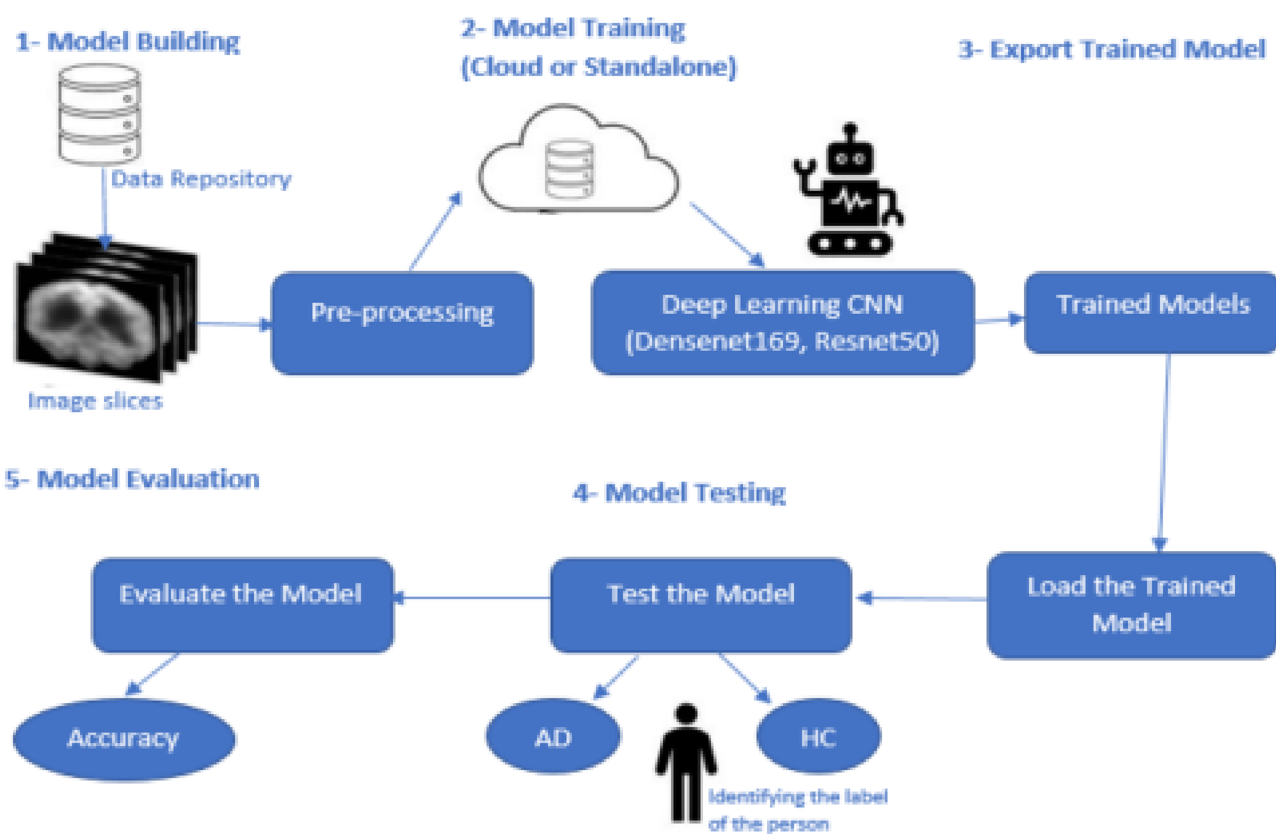
Traditional Methods

Traditional methods for brain tumor segmentation often rely on handcrafted features and machine learning algorithms. These approaches include thresholding, region-growing, and clustering techniques. While effective to some extent, these methods often struggle with complex tumor shapes and variations in image quality.

Deep Learning Approaches

Deep learning has emerged as a powerful tool for medical image segmentation, offering the ability to automatically learn features from data. Convolutional Neural Networks (CNNs) have shown promising results in various medical imaging tasks, including brain tumor segmentation. Notable architectures such as U-Net, ResNet, and DenseNet have been adapted and customized for this purpose.

3. SYSTEM ARCHITECTURE



EffNetB0, InceptionV3

4. PROJECT IMPLEMENTATION

PROPOSED METHODOLOGY

This proposed research methodology addresses the problems discussed in the Introduction. Various techniques based on deep learning were discussed earlier, but these approaches are lacking in the early diagnosis of AD when symptoms are minor or non-existent. This research study focused on the CNN-based deep learning models known as EfficientNetB0, InceptionV3 and Custom CNN, which are used for diagnosing and classifying AD.

Pre-processing

Several steps are performed in this stage of model development. The first is loading the image dataset into the model. The dataset is insufficient to train the deep learning models to meet the required data volume. Here data augmentation techniques are applied in which a few parameters are set such that the images are rescaled, rotated, zoomed, flipped horizontally and vertically, and split. Validation is performed for the whole image dataset. After applying these steps, sufficient data volume was generated based on the previous dataset. This dataset had two classes: Tumor and non-Tumor. Each image was individually labelled for analysis purposes. After applying these steps, the dataset was available for further processing.

Model building

The proposed model was based on the UNet, RESNET and Custom CNN models. CNN is a deep learning model used for the classification of features. A CNN model contains several different layers. A few common layers in CNN architectures are discussed below.

- **The input layer**

  The first CNN layer is the input layer which defines the image size used in the dataset as 128x128x3(width height channels). Shuffling the images is unnecessary because each training epoch will be shuffled automatically.

- **The convolutional layer**

  The convolutional layer is the core of the CNN architecture. It is the basic master layer which contains required parameters and feature maps. These layers are key to performance via the selection of a kernel. Padding is used in the feature maps and the convolutional layer to match the sizes of the input and output layers. By default, padding is assigned preset values of one.

- **Batch normalization layer**

  Network training is a time-consuming process. Batch normalization of the training data is an easy optimization. Gradients are normalized and help move forward or trigger the network propagation. Batch normalization layers are used between the non-linearity and convolutional layers.

- **ReLU layer**

  Rectified linear unit (ReLU) is the activation function mostly used in neural network architectures. ReLU layer is used after the batch normalization layer.

- **Max-pooling layer**

  Spatial features are large. This layer helps to reduce the feature map size and to remove redundancy from the spatial information. Sample reduction helps to reduce the computation cost and to move meaningful information into the feature maps.

- **Fully connected layer**

  These layers are described by their name. All the previous layers are connected here. At this stage, all the previous learning layers are merged. The final layer provides the classification of the model. The number of outputs equals the number of classes identified in the image dataset. In this study, there are four classes.

- **Softmax layer**

  The output obtained from the fully connected layer is not normalized. Normalization is performed using an output Softmax layer. The output obtained from this layer is a positive integer and can be used for classification.

- **Classification layer**

  The classification layer is the last in the CNN architecture. The Softmax layer uses it for classification. Probabilities are returned for each input image to authenticate the manually labelled classes. It also calculates the loss values.

Model training

After successfully developing the deep learning model architecture, it needs to be trained. The whole dataset is shuffled in each training epoch for 100. In this study, both architectures, DenseNet169 and ResNet50, were used separately for training purposes. We used 70% of the data for training the model and the rest of the 30% was used to test the model.

Testing model

The trained model is tested using various images. This model diagnoses AD and classification between these four different classes. The values output from the model are compared with the true values for the input images. The comparison of output values with true values for the testing dataset is used to evaluate the model.

Model evaluation

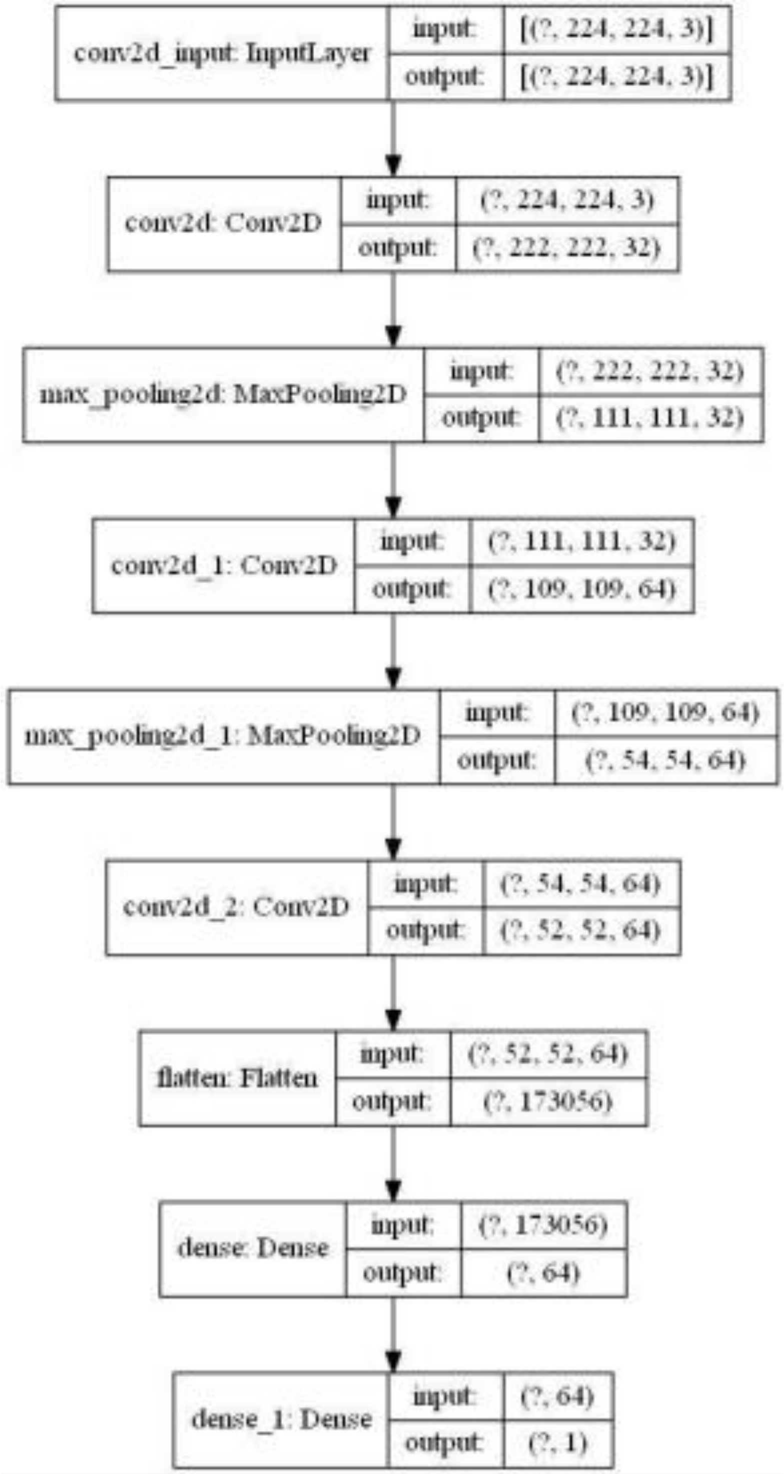
The model is evaluated based on the training model and testing dataset. The evaluation measure is computed by comparing calculated values from the model and true values known for each image in the testing dataset. The evaluation measures are used as proof of whether the model is performing well or not. Accuracy is the evaluation measure used to check the model's significance. Accuracy is used for the performance analysis of the proposed model. The computation of accuracy is defined below:

Accuracy = Aa/ Ac * 100

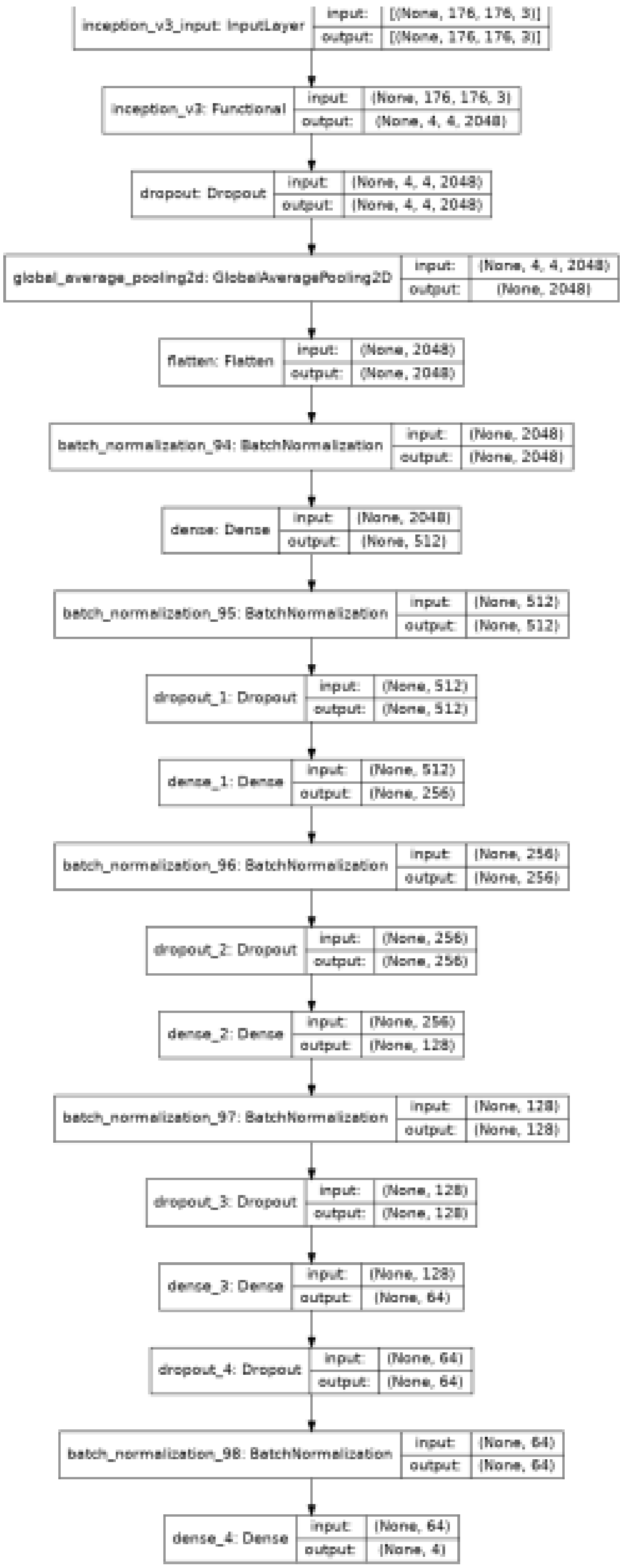
where Aa is the number of accurately classified results and Ac is the total number of results.

Algorithm

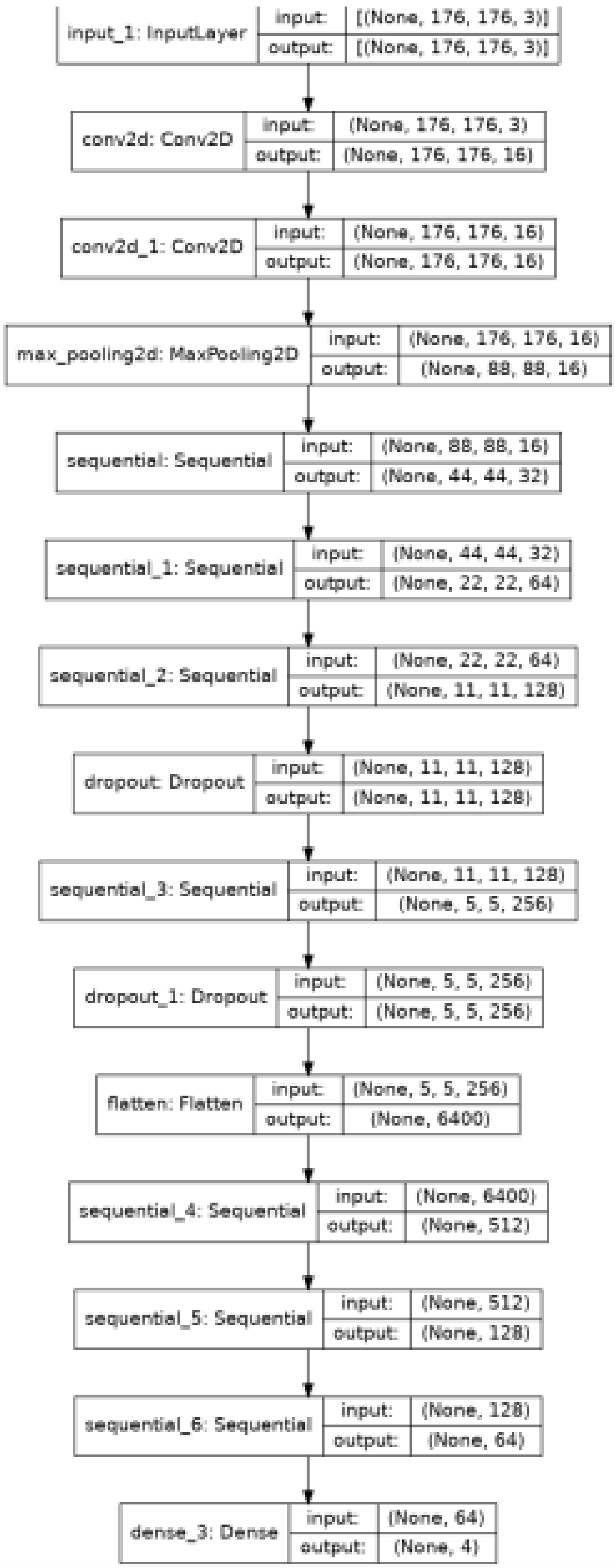
1. **EfficientNetB0**



2. InceptionV3


inception_v3_input: InputLayer
input: [(None, 176, 176, 3)]
output: [(None, 176, 176, 3)]
inception_v3: Functional
input: (None, 176, 176, 3)
output: (None, 4, 4, 2048)
dropout: Dropout
input: (None, 4, 4, 2048)
output: (None, 4, 4, 2048)
global_average_pooling2d: GlobalAveragePooling2D
input: (None, 4, 4, 2048)
output: (None, 2048)
flatten: Flatten
input: (None, 2048)
output: (None, 2048)
batch_normalization_94: BatchNormalization
input: (None, 2048)
output: (None, 2048)
dense: Dense
input: (None, 2048)
output: (None, 512)
batch_normalization_95: BatchNormalization
input: (None, 512)
output: (None, 512)
dropout_1: Dropout
input: (None, 512)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 256)
batch_normalization_96: BatchNormalization
input: (None, 256)
output: (None, 256)
dropout_2: Dropout
input: (None, 256)
output: (None, 256)
dense_2: Dense
input: (None, 256)
output: (None, 128)
batch_normalization_97: BatchNormalization
input: (None, 128)
output: (None, 128)
dropout_3: Dropout
input: (None, 128)
output: (None, 128)
dense_3: Dense
input: (None, 128)
output: (None, 64)
dropout_4: Dropout
input: (None, 64)
output: (None, 64)
batch_normalization_98: BatchNormalization
input: (None, 64)
output: (None, 64)
dense_4: Dense
input: (None, 64)
output: (None, 4)

3. Custom CNN



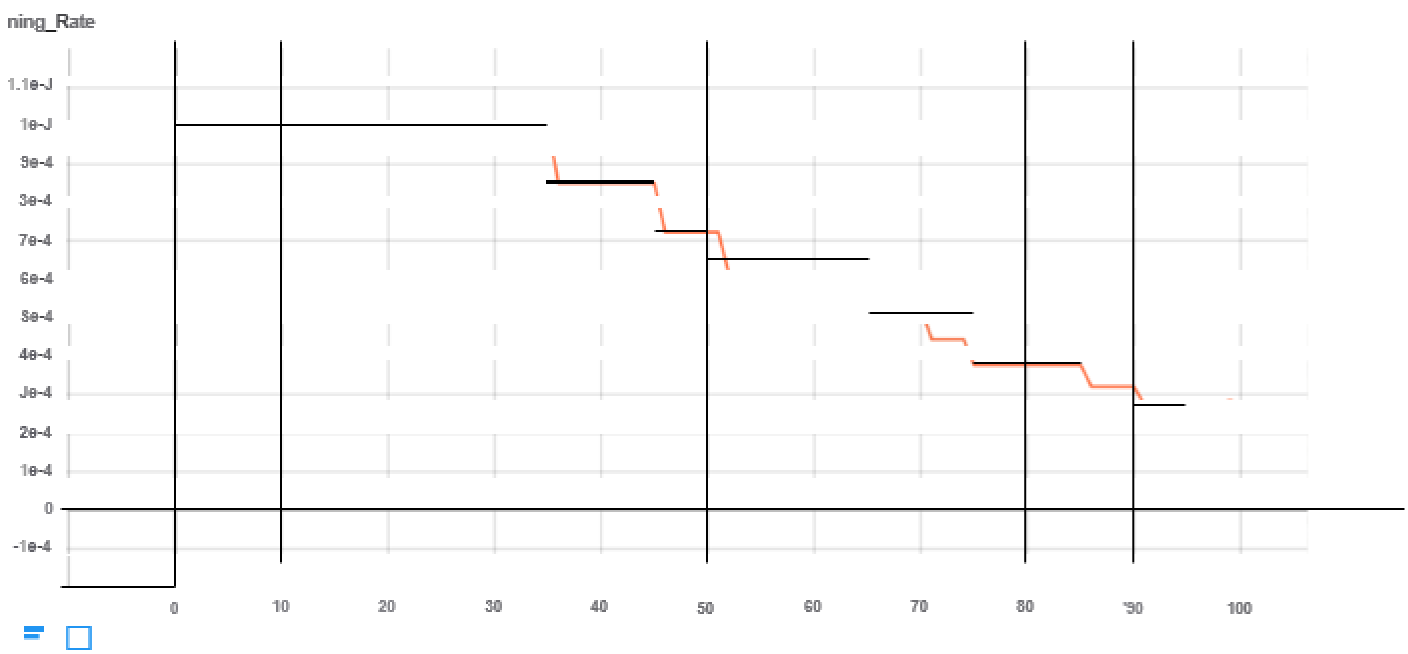
5. RESULT

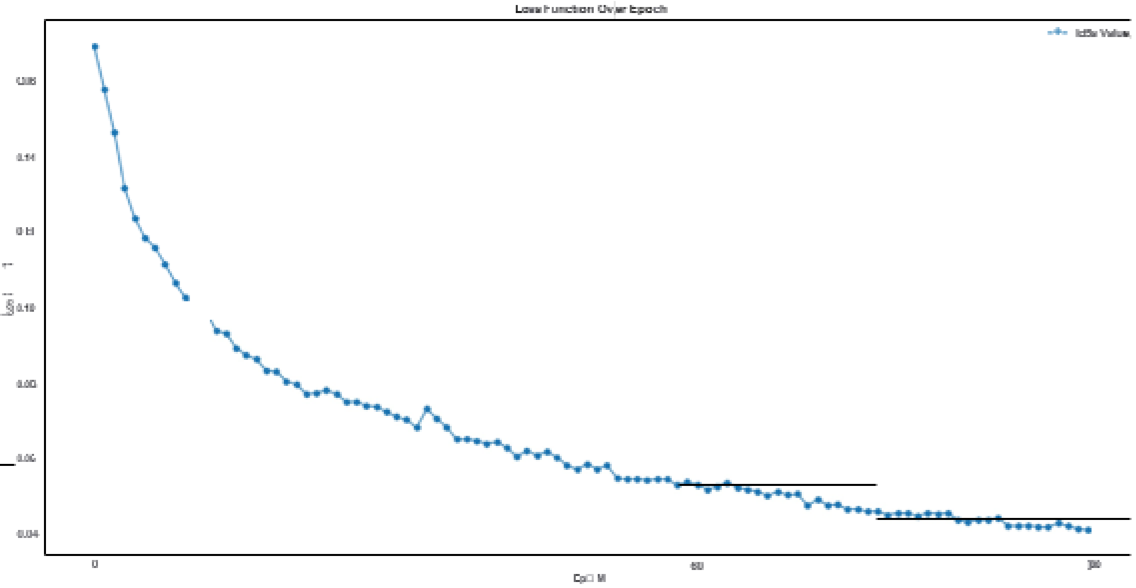
Outcome

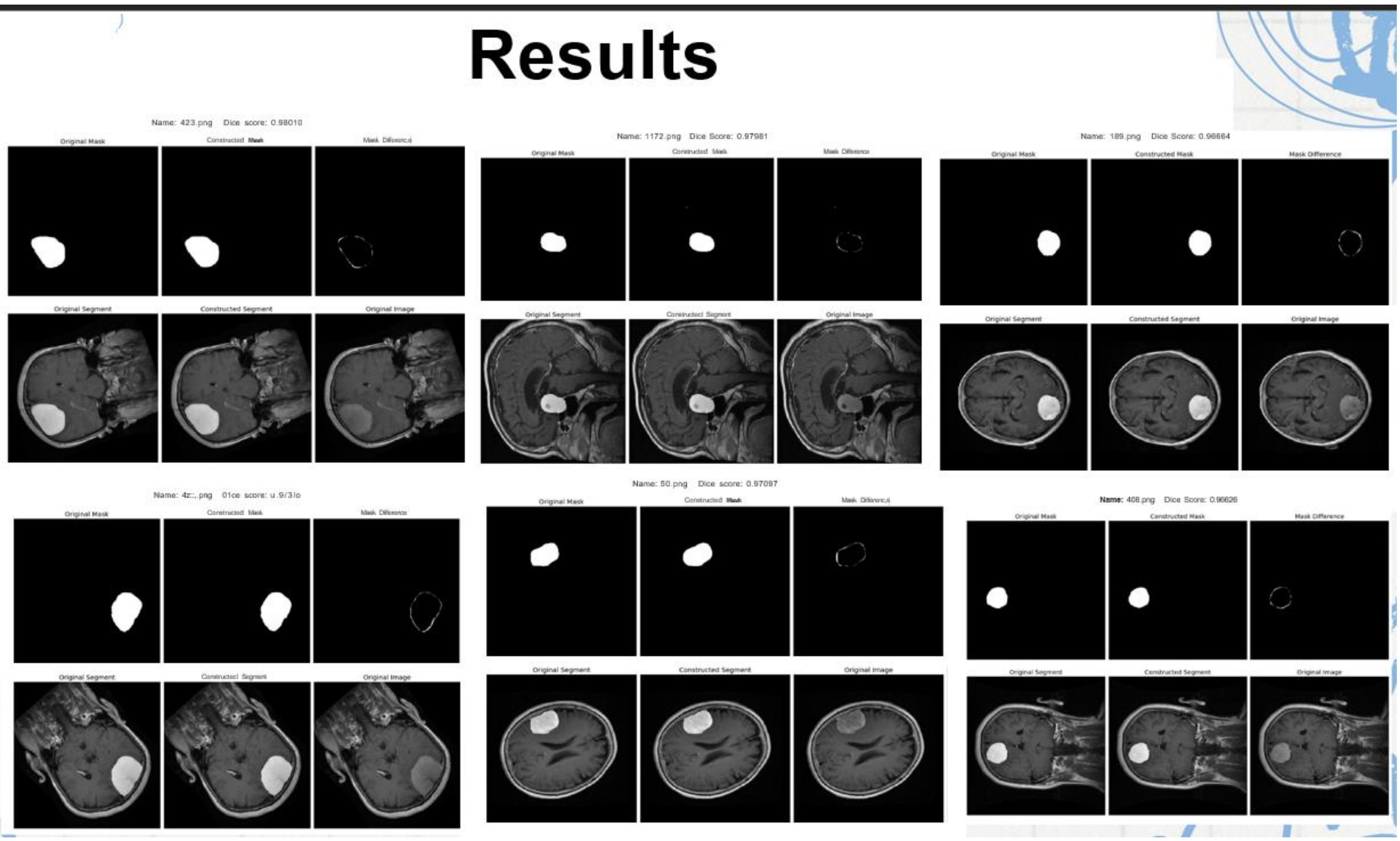
Model	Accuracy (%)
EfficientNetB0	**85.46**

InceptionV3	**89.06**
Custom CNN	**94.84**

Screenshots







6. CONCLUSION AND FUTURE WORK

It has been determined that Alzheimer's disease is an incurable neurodegenerative disease that affects brain memory, particularly in the elderly. Owing to the enormous number of patients, it is impossible to perform manual diagnosis efficiently and health specialists make errors during evaluation due to time constraints and the difficulty of the process. Various procedures are used to diagnose and characterize Alzheimer's, but an accurate and timely diagnostic solution is required.

The proposed model suggests a deep learning-based method for diagnosing and classifying Alzheimer's disease utilizing the EfficientNetB0, InceptionV3 and Custom CNN architectures. Non-Dementia, Very Mild-Dementia, Mild Dementia, and Moderate Dementia were the four classifications of Alzheimer's Disease in this model. During the training and testing stages, the Custom CNN method outperformed all other. This suggested approach may be used to do real-time analysis and classification of Alzheimer's disease. In the future, we plan to extend the disease detection with more data sets and use different measures to detect the system's accuracy.